AF
IIW

FEE TRANSMITTAL for FY 2004

Patent fees are subject to annual revision.

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT | $ 330

Complete if Known	
Application Number	09/810,167
Filing Date	3/19/2001
First Named Inventor	Kiernan et al.
Examiner Name	Hung Q. Pham
Art Unit	2172
Attorney Docket No.	ARC920010026US1

METHOD OF PAYMENT

☐ Check ☐ Credit Card ☐ Money Order ☐ Other

☒ Deposit Account:

Deposit Account Number: 09-0441

Deposit Account Name: IBM CORPORATION

The Director is authorized to: *(check all that apply)*

☒ Charge fee(s) indicated below ☒ Credit any overpayments

☒ Charge any additional fee(s) or any underpayment of fee(s)

☐ Charge fee(s) indicated below, **except for the filing fee** to the above-identified deposit account.

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1001	770	2001	385	Utility filing fee	
1002	340	2002	170	Design filing fee	
1003	530	2003	265	Plant filing fee	
1004	770	2004	385	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) $ 0

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

		Extra Claims		Fee from below	Fee Paid
Total Claims	-20** =		X		=
Independent Claims	-3** =		X		=
Multiple Dependent					=

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	86	2201	43	Independent claims in excess of 3
1203	290	2203	145	Multiple dependent claim, if not paid
1204	86	2204	43	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) $ 0

**or number previously paid, if greater; For Reissues, see above

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
1051	130	2051	65	Surcharge – late filing fee or oath	
1052	50	2052	25	Surcharge – late provisional filing fee or cover sheet	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for *ex parte* reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	110	2251	55	Extension for reply within first month	
1252	420	2252	210	Extension for reply within second month	
1253	950	2253	465	Extension for reply within third month	
1254	1,480	2254	740	Extension for reply within fourth month	
1255	2,010	2255	1,005	Extension for reply within fifth month	
1401	330	1401	165	Notice of Appeal	
1402	330	2402	165	Filing a brief in support of an appeal	330
1403	290	2403	145	Request for oral hearing	
1451	1,510	1452	1,510	Petition to institute a public use proceeding	
1452	110	2452	55	Petition to revive – unavoidable	
1453	1,330	2453	665	Petition to revive – unintentional	
1501	1,330	2501	665	Utility issue fee (or reissue)	
1502	480	2502	240	Design issue fee	
1503	640	2503	320	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	
1809	770	2809	385	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	770	2810	385	For each additional invention to be examined (37 CFR 1.129(b))	
1801	770	2801	385	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	
				Other fee (specify):	

*Reduced by Basic Filing Fee Paid SUBTOTAL (3) $330

Name (Print/Type)	Randy W. Lacasse	Registration No. (Attorney/Agent)	34,368	Telephone	703-838-7683
Signature	R. W. Lacasse			Date	August 13, 2004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPEAL BRIEF – 37 C.F.R. § 1.192

U.S. Patent Application 09/810,167 entitled,

"Tagging XML Query Results Over Relational DBMSs"

Real Party in Interest: International Business Machines Corporation

Related Appeals and Interferences:

None

Status of Claims:

Claims 1-46 are pending.

Claims 1, 3-4, 10-13, 18-22, 24-25, 31-34, and 39-45 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Fernandez et al. (USP 6,604,100).

Claims 2, 5-9, 14-17, 23, 26-30, 25-38, and 46 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Status of Amendments:

Amendment filed 3/4/2004 was entered.

Summary of the Invention:

The presently claimed invention provides a method of translating queries of XML data into queries against a relational database. In addition, a method is provided for publishing the queried relational data as an XML document thus precluding the necessity for an intermediate data representation.

Translation occurs when an XML query is parsed and transformed into an intermediate language-neutral query representation. The intermediate language-neutral query representation is then transformed into an SQL query that is issued over a relational table in the database where the relational table corresponds to a virtual XML document.

A tagger graph is generated from the intermediate language-neutral representation of the XML query. Each of the nodes of the tagger graph is an operator that performs processing on the results of the SQL query results. Traversing the tagger graph generates tags and structure for XML output. When SQL query results from the relational database are provided as input to the tagger graph and a traversal of the graph generates the tags and the structure by processing these inputs, a structured XML document is produced.

Based on the structure of the initial XML query, a tagger graph is generated during run-

time to put currently flattened relational tuples back in a hierarchical graphical XML structure.

Pending Claims (all citations are made from the original specification, including the figures):

1. A method of tagging results of an XML query over a relational database, said method comprising:

generating a tagger tree graph from said XML query ***(page 17, lines 3-4)***, each node of said tagger tree graph ***(page 17, lines 8-9)*** comprising a tagger operator ***(page 17, lines 8-9)***, each tagger operator having a parse tree associated therewith ***(page 21, lines 8-9)***;

calling each tagger operator in accordance with a structure of said tagger tree graph ***(page 22, lines 12-13)***, and

evaluating said parse trees associated with each called tagger operator to tag results of said XML query over said relational database ***(page 22, lines 18-20; and element 1108)***.

2. A method of tagging results of an XML query over a relational database, as per claim 1, wherein said tagger node graph has a top-most tagger operator and a plurality of lower-most tagger operators ***(page 17, lines 1-2)***, said calling and evaluating steps further comprising:

a. starting with said top-most tagger operator, each tagger operator implementing a method to request results from inputs to said tagger operator, said method causing lower-level tagger operators connected to said inputs to be called ***(page 22, lines 14-16)***;

b. starting with said lower-most tagger operators, each called tagger operator returning intermediate tagged results to a higher-level connected tagger operator upon evaluating said associated parse tree ***(page 22, lines 14-16; and page 30, line 21)***;

performing steps a and b until an end of said results of said XML query is reached ***(page 22, starting on line 16***, and

said top-most tagger operator producing tagged output XML of said results of said XML query ***(page 23, lines 1-3)***.

3. A method of tagging results of an XML query over a relational database, as per claim 1,

wherein said tagger operators comprise any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator ***(page 20, line 1)***.

4. A method of tagging results of an XML query over a relational database, as per claim 1, wherein said tagger tree graph includes a tagger operator for each level in a result XML tree of said XML query ***(page 21, lines 6-7)***.

5. A method of tagging results of an XML query over a relational database, as per claim 4, wherein said tagger input operators execute in a sorted outer union mode ***(page 24, paragraph 3; and page 29, lines 3-4)***.

6. A method of tagging results of an XML query over a relational database, as per claim 5, wherein said tagger input operators comprise a shared tagger row stream ***(page 23, lines 7-9)***.

7. A method of tagging results of an XML query over a relational database, as per claim 4, wherein said tagger input operators execute in a node strip mode ***(page 26, lines 2-5; and page 29, lines 3-4)***.

8. A method of tagging results of an XML query over a relational database, as per claim 7, wherein each of said tagger operators comprises a tagger row stream ***(page 26, paragraph 1)***.

9. A method of tagging results of an XML query over a relational database, as per claim 1, wherein each tagger operator ***(element 1100)*** performs any of a cr8_elem, a cr8_attr, a cr8_attr_list, a cr8_fragments or a cr8_fragment_list function ***(page 30, paragraph 1)***.

10. A method of tagging results of an XML query over a relational database, as per claim 1, wherein each tagger operator implements a next method to produce a result row ***(page 31, lines 14-15)***.

11. A method of tagging results of an XML query over a relational database, as per claim 1,

said method further comprising:

parsing said XML query ***(figure 14, element 1400)***;

transforming said XML queries into a language-neutral intermediate representation ***(figure 14, element 1402)***;

rewriting said language-neutral intermediate representation into an equivalent form easily translated into an SQL query ***(figure 14, element 1404; and page 15, line 3)***;

translating said equivalent form into one or more SQL queries over said relational database ***(figures 14, element 1406)***, and

executing said one or more SQL queries to produce said results of said XML query over said relational database ***(figure 14, element 1408)***.

12. A method of tagging results of an XML query over a relational database, as per claim 11, wherein said tagger tree graph is generated from said equivalent form ***(page 17, lines 7-8; and page 15, lines 18-20)***.

13. A method of tagging results of an XML query over a relational database, as per claim 11, wherein said tagger tree graph includes a tagger operator for each node in a result XML tree of said XML query ***(page 17, lines 8-10)***.

14. A method of tagging results of an XML query over a relational database, as per claim 13, wherein said tagger input operators execute in a sorted outer union mode and said translating step produces a single SQL query ***(page 23, line 7)*** to produce a single sorted outer union relational database result ***(figures 12a and 12f; page 23, paragraph 3; and page 24, paragraph 4)***.

15. A method of tagging results of an XML query over a relational database, as per claim 14, wherein said tagger input operators comprise a shared tagger row stream ***(page 23, lines 7-9)***.

16. A method of tagging results of an XML query over a relational database, as per claim 13, wherein said tagger input operators execute in a node strip mode and said translating step

produces a set of SQL queries to produce a set of node strip relational database results ***(page 26, line 13)***.

17. A method of tagging results of an XML query over a relational database, as per claim 16, wherein each of said tagger operators comprises a tagger row stream ***(page 26, paragraph 1)***.

18. A method of tagging results of an XML query over a relational database, as per claim 11, wherein said tagger operators comprise any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator ***(page 20, line 1)***.

19. A method of tagging results of an XML query over a relational database, as per claim 11, wherein a number of relational database tables of said relational database are mapped to a number of virtual XML documents and said XML queries are issued over said virtual XML documents ***(figure 14, element 1400; and page 32, lines 3-5)***.

20. A method of tagging results of an XML query over a relational database, as per claim 1, wherein said method operates over a distributed computing network ***(figure 1, element 120; and page 8, lines 3-4***.

21. A method of tagging results of an XML query over a relational database, as per claim 20, wherein said method operates over the Internet ***(page 8, lines 3-4; and page 9, lines 9-10)***.

22. A system for tagging results of an XML query over a relational database, said system comprising:

a tagger runtime ***(figure 2, element 240)***;

a tagger tree graph generated from said XML query, each node of said tagger tree graph comprising a tagger operator ***(page 17, lines 3-4 and 8-9)***;

a parse tree associated with each tagger operator ***(page 17, lines 8-9)***, and

wherein said tagger runtime calls each tagger operator in accordance with a structure ***(page 28, lines 3-5)*** of said tagger tree graph and evaluates said parse trees associated with each called

tagger operator to tag results of said XML query over said relational database ***(page 28, lines 6-8)***.

23. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said tagger node graph has a top-most tagger operator and a plurality of lower-most tagger operators, and to perform said calling and evaluating, said tagger runtime further:

a. starting with said top-most tagger operator, causing each tagger operator to implement a method to request results from inputs to said tagger operator, said method causing lower-level tagger operators connected to said inputs to be called ***(page 28, lines 3-5 and 9-10)***;

b. starting with said lower-most tagger operators, causing each called tagger operator to return intermediate tagged results to a higher-level connected tagger operator upon evaluating said associated parse tree ***(page 30, lines 16-21)***;

performing steps a and b until an end of said results of said XML query is reached ***(page 28, lines 19-20)***, and

upon reaching an end of said results of said XML query, causing said top-most tagger operator to produce a tagged output XML document of said results of said XML query ***(page 29, lines 1-2)***.

24. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said tagger operators comprise any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator ***(page 20, line 1)***.

25. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said tagger graph includes a tagger input operator for each node in a result XML tree of said XML query ***(page 21, line 6-7)***.

26. A system for tagging results of an XML query over a relational database, as per claim 25, wherein said tagger input operators execute in a sorted outer union mode ***(page 24, paragraph 3; and page 29, lines 3-4)***.

27. A system for tagging results of an XML query over a relational database, as per claim 26,

wherein said tagger input operators comprise a shared tagger row stream ***(page 23, lines 7-9)***.

28. A system for tagging results of an XML query over a relational database, as per claim 25, wherein said tagger input operators execute in a node strip mode ***(page 26, lines 2-5; page 29, lines 3-4)***.

29. A system for tagging results of an XML query over a relational database, as per claim 28, wherein each of said tagger operators comprises a tagger row stream ***(page 26, paragraph 7)***.

30. A system for tagging results of an XML query over a relational database, as per claim 22, wherein each tagger operator performs any of a cr8_elem, a cr8_attr, a cr8_attr_list, a cr8_fragments or a cr8_fragment_list function ***(page 30, paragraph 1)***.

31. A system for tagging results of an XML query over a relational database, as per claim 22, wherein each tagger operator implements a next method to produce a result row ***(page 31, lines 14-15)***.

32. A system for tagging results of an XML query over a relational database, as per claim 22, said system further comprising:

a parser, said parser parsing said XML query and transforming said XML queries into a language-neutral intermediate representation ***(figure 2, element 210; and page 10, paragraph 4)***;

a rewrite engine, said rewrite engine rewriting said language-neutral intermediate representation into an equivalent form easily translated into an SQL query ***(figure 2, element 220; and page 11, lines 1-2)***;

a translator, said translator translating said equivalent form into one or more SQL queries over said relational database ***(figure 2, element 230; and page 11, lines 3-5)***, and

an RDBMS, said RDBMS executing said one or more SQL queries to produce said results of said XML query over said relational database ***(page 11, line 7)***.

33. A system for tagging results of an XML query over a relational database, as per claim 32,

wherein said tagger graph is generated from said equivalent form ***(page 17, lines 7-8; and page 15, lines 18-20)***.

34. A system for tagging results of an XML query over a relational database, as per claim 32, wherein said tagger graph includes a tagger input operator for each node in a result XML tree of said XML query ***(page 17, lines 8-10)***.

35. A system for tagging results of an XML query over a relational database, as per claim 34, wherein said tagger input operators execute in a sorted outer union mode and said translator produces a single SQL query ***(page 23, line 7)*** to produce a single sorted outer union relational database result ***(figures 12a and 12f; page 23, paragraph 3; and page 24, paragraph 4)***.

36. A system for tagging results of an XML query over a relational database, as per claim 35, wherein said tagger input operators comprise a shared tagger row stream ***(page 23, lines 7-9)***.

37. A system for tagging results of an XML query over a relational database, as per claim 34, wherein said tagger input operators execute in a node strip mode and said translator produces a set of SQL queries to produce a set of node strip relational database results ***(page 26, line 13)***.

38. A system for tagging results of an XML query over a relational database, as per claim 37, wherein each of said tagger operators comprises a tagger row stream ***(page 26, paragraph 1)***.

39. A system for tagging results of an XML query over a relational database, as per claim 32, wherein said tagger operators comprise any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator ***(page 20, line 1)***.

40. A system for tagging results of an XML query over a relational database, as per claim 32, said system further comprising:

a schema mapper ***(figure 2, element 200)***, said schema mapper mapping a number of relational database tables of said relational database to a number of virtual XML documents ***(page 10, lines***

14-15), and

an XML-QL engine, said XML-QL engine issuing said XML queries over said virtual XML documents ***(page 10, lines 15-16)***.

41. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said system operates over a distributed computing network ***(figure 1, element 120; and page 8, lines 3-4)***.

42. A system for tagging results of an XML query over a relational database, as per claim 41, wherein said system operates over the Internet ***(page 8, lines 3-4; and page 9, lines 9-10)***.

43. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said tagger runtime operates outside an RDBMS.

44. A system for tagging results of an XML query over a relational database, said system comprising:

means for generating a tagger tree graph from said XML query, each node of said tagger tree graph comprising a tagger operator, each tagger operator having a parse tree associated therewith ***(page 17, paragraph 1; and page 21, paragraph 2)***;

means for calling each tagger operator in accordance with a structure of said tagger tree graph ***(page 17, lines 8 and 10)***, and

means for evaluating said parse trees associated with each called tagger operator to tag results of said XML query over said relational database ***(page 28, lines 6-7)***.

45. A computer program product comprising a machine-readable medium including computer readable program code therein for tagging results of an XML query over a relational database comprising:

computer readable program code generating a tagger tree graph from said XML query, each node of said tagger tree graph comprising a tagger operator, each tagger operator having a parse tree associated therewith ***(page 17, paragraph 1; and page 21, paragraph 2)***;

computer readable program code calling each tagger operator in accordance with a structure of said tagger tree graph ***(page 17, lines 8 and 10)***, and

computer readable program code evaluating said parse trees associated with each called tagger operator to tag results of said XML query over said relational database ***(page 28, lines 6-7)***.

46. A computer program product comprising a machine-readable medium including computer readable program code therein for tagging results of an XML query over a relational database as per claim 45, wherein said generated tagger node graph has a top-most tagger operator and a plurality of lower-most tagger operators, said calling and evaluating computer readable program code further comprising:

computer readable program code for performing:

a. starting with said top-most tagger operator, each tagger operator requesting results from inputs to said tagger operator, said request causing lower-level tagger operators connected to said inputs to be called ***(page 28, lines 3-5 and 9-10)***;

b. starting with said lower-most tagger operators, each called tagger operator returning intermediate tagged results to a higher-level connected tagger operator upon evaluating said associated parse tree ***(page 30, lines 16-21)***;

performing steps a and b until an end of said results of said XML query is reached ***(page 28, lines 19-20)***, and

said top-most tagger operator producing tagged output XML of said results of said XML query ***(page 29, lines 1-2)***.

Issues:

I. Was a proper rejection made under 35 U.S. C. § 103(a) using existing USPTO guidelines?

Grouping of Claims:

All claims stand or fall together (1-46).

Argument:

I. REJECTIONS UNDER 35 U.S.C. § 103(a)

To establish a prima facie case of obviousness under U.S.C. § 103, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. Additionally, the teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, and not based on applicant's disclosure (In re Vaeck, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991)).

Claims 1, 3, 4, 10-13, 18-22, 24, 25, 31-34, and 39-45 are rejected under 35 U.S.C. 103(a) as being unpatentable over Fernandez et al. [USP 6,604,100].

The Fernandez et al. reference (hereafter Fernandez) provides for a method to convert an XML-QL user query over an XML view to a Relational to XML transformation language (RXL) query, convert an RXL query to an SQL query, and retrieve relational data stored in a relational database. By contrast, the present invention provides for the tagging of relational data resultant from the execution of an XML query over a relational database. With regards to claims 1, 22, 44, and 45, the examiner has cited the Fernandez reference, stating that conversion from relational data to XML is taught. However, a closer reading of the sections referenced by the examiner in column 3, lines 10-24 and in col. 4, line 64 through col. 5, line 25 indicate an XML-QL user query formulated on an initial XML virtual view query, the execution of which results in an XML document. No mention is made of generating a tagger tree graph, nor for tagging the

results of XML-QL user query execution to produce such an XML document. Furthermore, along with each instance of an XML Generator Module **106** that is mentioned in conjunction with a resultant XML document in Fernandez, accompanying discussion describes SQL query results as being "merged to construct XML output". More the specifically, *the Fernandez method "plugs" SQL query results into a predetermined XML template that is known prior to query execution, in order to produce an XML document*, as described in column 5, lines 25-35. On the other hand, the present invention generates a tagger tree graph from an XML query in order to tag the results of an XML query execution over a relational database, such that these results display an XML document. In addition, the present invention tags results of an XML query execution in accordance with the structure of a generated tagger tree graph. Because tagging these results depends on the structure of a generated tagger tree graph, ***the structure of XML output of the present invention is determined subsequent to query execution.***

The examiner has equated "view query v", as disclosed by Fernandez, with a tagger tree graph of the present invention, by citing column 12, lines 32-45 and column 12, lines 60-61. A closer reading of the cited sections indicates that a view query template conforming to an initial XML virtual view of a relational database may be represented by a data structure called a view tree. In contrast, a tagger tree graph of the present invention does not conform to nor does it represent any one particular XML view; rather, a distinct tagger tree graph is generated for each individual XML query issued over a relational database. The examiner has also pointed to column 21, lines 43-48, as disclosing a tagger operator, and a parse tree. However, the cited sections in column 12 and 21 indicate a view tree used in the formulation of a relational database query, not in the structured tagging of relational data resulting from the execution of an XML query, as with the present invention. The view tree representing view query v as disclosed by Fernandez is the basis for the compile-time composition of a query, C, to be executed over a relational database (see column 12, lines 44-48 and lines 60-61), ***whereas the present invention discloses the opposite, a tagger tree graph generated based on an XML query over a relational database.***

As stated by the examiner, Fernandez does not explicitly teach the step of calling each tagger operator in accordance with a structure of said tagger tree graph, nor does he teach evaluating said parse trees associated with each called tagger operator to tag results of said XML

query over said relational database. *Indeed, Fernandez cannot possibly teach the calling of each tagger operator in accordance with a structure of said tagger tree graph, nor the evaluation of parse trees associated therewith because Fernandez discloses neither the generation of a tagger tree graph nor a tagger runtime.* Because the view tree representation of view query V is utilized to create the composed query C that has not yet been executed, no SQL results are available to the view tree, neither for calling a tagger operator, nor for evaluating a parse tree. In stark contrast, the present invention teaches not only a tagger tree operator called in an order corresponding to the structure of the tagger tree graph to which it belongs, but also the evaluation of a parse tree that is associated with each tagger operator for the purpose of tagging relational data resulting from the execution of the XML query over a relational database.

A closer reading of the examiner-cited passages in column 22, lines 53-56, and column 23, lines 20-35 reveals that the view tree simply serves as a current context for which a list of all possible RXL block solutions are enumerated for rewriting an XML-QL query over the current view, prior to execution (see column 22, lines 64-66). In requiring RXL blocks to be enumerated when evaluated on the current view tree (see column 22, lines 63-65), the Fernandez method is limited to outputting only resulting XML documents that are structured in accordance with the initial XML virtual view. As opposed to enumerating all, the present invention teaches a tagger runtime calling each tagger operator in a generated tagger tree graph to tag the results of an XML query over a relational database. On the other hand, the present invention discloses a method for tagging results of in accordance with the structure of a tagger tree graph that is generated from an XML query. The examiner's citation of the Fernandez illustrative example in column 9, lines 61-65; simply serves to further strengthen the previous arguments; Fernandez describes only a mention of computing a view on demand is made, no enablement is detailed.

The referenced sections therefore fails to provide the claimed tagger tree graph, tagger tree operator, and parse tree associated and make no mention of using a tagger graph to tag and structure the results of an XML query over a relational database, and thus fails to provide a method for evaluating parse trees associated with each tagger operator to tag of said XML language over said relational database in the manner of the present invention. The examiner has recited the exact language of the remaining claims (3-4, 10-13, 18-21, 24-25, 31-34, and 39-43) and has suggested a correlation with the Fernandez reference. However, the examiner has not

established the required *prima facia* case of obviousness. Specifically, the examiner has not set forth any elements in Fernandez that provide for the claimed features. It is our position that Fernandez does not describe or suggest the following claimed features.

With regards to dependent claims 3, 4, 13, 18, 24, and 25, 34, and 39, the examiner's argument that the Fernandez reference provides for tagger operators comprising any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator is moot, as is the argument that the Fernandez reference provides for a tagger tree graph including a tagger operator for each level in a result XML tree of said XML query and a tagger input operator for each node in a result XML tree of said XML because Fernandez does not provide for a tagger tree graph to tag results of an XML query over a relational database, as is argued for independent claims 1, 22, 44, and 45.

With regards to dependent claims 10, 12, 31, and 33, Fernandez discloses neither generating said tagger tree graph in said equivalent form, nor does it disclose each tagger operator implements a next method to produce a result row because, as argued previous for independent claims 1, 22, 44, and 45, the Fernandez reference does not provide for generating a tagger tree graph.

With regards to dependent claims 11, 19, 20, 21, and 43, Fernandez makes no mention of tagging results of an XML query; therefore there is no basis to the argument examiner's argument that results are tagged by: parsing said XML query; transforming said XML queries into a language-neutral intermediate representation; rewriting said language-neutral intermediate representation into an equivalent form easily translated into an SQL query; translating said equivalent form into one or more SQL queries over said relational database, and executing said one or more SQL queries to produce said results of said XML query over said relational database, nor a number of relational database tables of said relational database are mapped to a number of virtual XML documents and said XML queries are issued over said virtual XML documents. Furthermore, there is no basis for the examiner's argument that Fernandez also provides for the tagging of results operating over a distributed computing network or the Internet; nor for the argument that said tagger runtime operates outside an RDBMS.

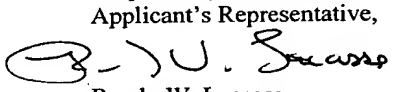
With regards to dependent claims 32, 40, 41, and 42, Fernandez makes no mention of tagging the results of an XML query over a relational database; hence, there is no basis for the

SUMMARY

As has been detailed above, none of the references, cited or applied, provide for the specific claimed details of applicant's presently claimed invention, nor render them obvious. It is believed that this case is in condition for allowance and reconsideration thereof and early issuance is respectfully requested.

A petition for extension of time is being filed with this Appeal Brief. The Commissioner is hereby authorized to charge any deficiencies in the fees provided, or to credit any overpayment, to Deposit Account No. 12-0010.

Respectfully submitted by
Applicant's Representative,



Randy W. Lacasse
Reg. No. 34,368

1725 Duke Street
Suite 650
Alexandria, VA 22314
(703) 838-7683

Appendix:

1. A method of tagging results of an XML query over a relational database, said method comprising:

generating a tagger tree graph from said XML query, each node of said tagger tree graph comprising a tagger operator, each tagger operator having a parse tree associated therewith;

calling each tagger operator in accordance with a structure of said tagger tree graph, and

evaluating said parse trees associated with each called tagger operator to tag results of said XML query over said relational database.

2. A method of tagging results of an XML query over a relational database, as per claim 1, wherein said tagger node graph has a top-most tagger operator and a plurality of lower-most tagger operators, said calling and evaluating steps further comprising:

a. starting with said top-most tagger operator, each tagger operator implementing a method to request results from inputs to said tagger operator, said method causing lower-level tagger operators connected to said inputs to be called;

b. starting with said lower-most tagger operators, each called tagger operator returning intermediate tagged results to a higher-level connected tagger operator upon evaluating said associated parse tree;

performing steps a and b until an end of said results of said XML query is reached, and

said top-most tagger operator producing tagged output XML of said results of said XML query.

3. A method of tagging results of an XML query over a relational database, as per claim 1, wherein said tagger operators comprise any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator.

4. A method of tagging results of an XML query over a relational database, as per claim 1, wherein said tagger tree graph includes a tagger operator for each level in a result XML tree of said XML query.

5. A method of tagging results of an XML query over a relational database, as per claim 4, wherein said tagger input operators execute in a sorted outer union mode.

6. A method of tagging results of an XML query over a relational database, as per claim 5, wherein said tagger input operators comprise a shared tagger row stream.

7. A method of tagging results of an XML query over a relational database, as per claim 4, wherein said tagger input operators execute in a node strip mode.

8. A method of tagging results of an XML query over a relational database, as per claim 7, wherein each of said tagger operators comprises a tagger row stream.

9. A method of tagging results of an XML query over a relational database, as per claim 1, wherein each tagger operator performs any of a cr8_elem, a cr8_attr, a cr8_attr_list, a cr8_fragments or a cr8_fragment_list function.

10. A method of tagging results of an XML query over a relational database, as per claim 1, wherein each tagger operator implements a next method to produce a result row.

11. A method of tagging results of an XML query over a relational database, as per claim 1, said method further comprising:

parsing said XML query;

transforming said XML queries into a language-neutral intermediate representation;

rewriting said language-neutral intermediate representation into an equivalent form easily translated into an SQL query;

translating said equivalent form into one or more SQL queries over said relational database, and

executing said one or more SQL queries to produce said results of said XML query over said relational database.

12. A method of tagging results of an XML query over a relational database, as per claim 11, wherein said tagger tree graph is generated from said equivalent form.

13. A method of tagging results of an XML query over a relational database, as per claim 11, wherein said tagger tree graph includes a tagger operator for each node in a result XML tree of said XML query.

14. A method of tagging results of an XML query over a relational database, as per claim 13, wherein said tagger input operators execute in a sorted outer union mode and said translating step produces a single SQL query to produce a single sorted outer union relational database result.

15. A method of tagging results of an XML query over a relational database, as per claim 14, wherein said tagger input operators comprise a shared tagger row stream.

16. A method of tagging results of an XML query over a relational database, as per claim 13, wherein said tagger input operators execute in a node strip mode and said translating step produces a set of SQL queries to produce a set of node strip relational database results.

17. A method of tagging results of an XML query over a relational database, as per claim 16, wherein each of said tagger operators comprises a tagger row stream.

18. A method of tagging results of an XML query over a relational database, as per claim 11, wherein said tagger operators comprise any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator.

19. A method of tagging results of an XML query over a relational database, as per claim 11, wherein a number of relational database tables of said relational database are mapped to a number of virtual XML documents and said XML queries are issued over said virtual XML

documents.

20. A method of tagging results of an XML query over a relational database, as per claim 1, wherein said method operates over a distributed computing network.

21. A method of tagging results of an XML query over a relational database, as per claim 20, wherein said method operates over the Internet.

22. A system for tagging results of an XML query over a relational database, said system comprising:

a tagger runtime;

a tagger tree graph generated from said XML query, each node of said tagger tree graph comprising a tagger operator;

a parse tree associated with each tagger operator, and

wherein said tagger runtime calls each tagger operator in accordance with a structure of said tagger tree graph and evaluates said parse trees associated with each called tagger operator to tag results of said XML query over said relational database.

23. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said tagger node graph has a top-most tagger operator and a plurality of lower-most tagger operators, and to perform said calling and evaluating, said tagger runtime further:

a. starting with said top-most tagger operator, causing each tagger operator to implement a method to request results from inputs to said tagger operator, said method causing lower-level tagger operators connected to said inputs to be called;

b. starting with said lower-most tagger operators, causing each called tagger operator to return intermediate tagged results to a higher-level connected tagger operator upon evaluating said associated parse tree;

performing steps a and b until an end of said results of said XML query is reached, and

upon reaching an end of said results of said XML query, causing said top-most tagger operator to

produce a tagged output XML document of said results of said XML query.

24. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said tagger operators comprise any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator.

25. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said tagger graph includes a tagger input operator for each node in a result XML tree of said XML query.

26. A system for tagging results of an XML query over a relational database, as per claim 25, wherein said tagger input operators execute in a sorted outer union mode.

27. A system for tagging results of an XML query over a relational database, as per claim 26, wherein said tagger input operators comprise a shared tagger row stream.

28. A system for tagging results of an XML query over a relational database, as per claim 25, wherein said tagger input operators execute in a node strip mode.

29. A system for tagging results of an XML query over a relational database, as per claim 28, wherein each of said tagger operators comprises a tagger row stream.

30. A system for tagging results of an XML query over a relational database, as per claim 22, wherein each tagger operator performs any of a cr8_elem, a cr8_attr, a cr8_attr_list, a cr8_fragments or a cr8_fragment_list function.

31. A system for tagging results of an XML query over a relational database, as per claim 22, wherein each tagger operator implements a next method to produce a result row.

32. A system for tagging results of an XML query over a relational database, as per claim 22, said system further comprising:

a parser, said parser parsing said XML query and transforming said XML queries into a language-neutral intermediate representation;

a rewrite engine, said rewrite engine rewriting said language-neutral intermediate representation into an equivalent form easily translated into an SQL query;

a translator, said translator translating said equivalent form into one or more SQL queries over said relational database, and

an RDBMS, said RDBMS executing said one or more SQL queries to produce said results of said XML query over said relational database.

33. A system for tagging results of an XML query over a relational database, as per claim 32, wherein said tagger graph is generated from said equivalent form.

34. A system for tagging results of an XML query over a relational database, as per claim 32, wherein said tagger graph includes a tagger input operator for each node in a result XML tree of said XML query.

35. A system for tagging results of an XML query over a relational database, as per claim 34, wherein said tagger input operators execute in a sorted outer union mode and said translator produces a single SQL query to produce a single sorted outer union relational database result.

36. A system for tagging results of an XML query over a relational database, as per claim 35, wherein said tagger input operators comprise a shared tagger row stream.

37. A system for tagging results of an XML query over a relational database, as per claim 34, wherein said tagger input operators execute in a node strip mode and said translator produces a set of SQL queries to produce a set of node strip relational database results.

38. A system for tagging results of an XML query over a relational database, as per claim 37, wherein each of said tagger operators comprises a tagger row stream.

39. A system for tagging results of an XML query over a relational database, as per claim 32, wherein said tagger operators comprise any of a tagger input operator, a tagger scalar operator or a tagger aggregate operator.

40. A system for tagging results of an XML query over a relational database, as per claim 32, said system further comprising:

a schema mapper, said schema mapper mapping a number of relational database tables of said relational database to a number of virtual XML documents, and

an XML-QL engine, said XML-QL engine issuing said XML queries over said virtual XML documents.

41. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said system operates over a distributed computing network.

42. A system for tagging results of an XML query over a relational database, as per claim 41, wherein said system operates over the Internet.

43. A system for tagging results of an XML query over a relational database, as per claim 22, wherein said tagger runtime operates outside an RDBMS.

44. A system for tagging results of an XML query over a relational database, said system comprising:

means for generating a tagger tree graph from said XML query, each node of said tagger tree graph comprising a tagger operator, each tagger operator having a parse tree associated therewith;

means for calling each tagger operator in accordance with a structure of said tagger tree graph,

and

means for evaluating said parse trees associated with each called tagger operator to tag results of said XML query over said relational database.

45. A computer program product comprising a machine-readable medium including computer readable program code therein for tagging results of an XML query over a relational database comprising:

computer readable program code generating a tagger tree graph from said XML query, each node of said tagger tree graph comprising a tagger operator, each tagger operator having a parse tree associated therewith;

computer readable program code calling each tagger operator in accordance with a structure of said tagger tree graph, and

computer readable program code evaluating said parse trees associated with each called tagger operator to tag results of said XML query over said relational database.

46. A computer program product comprising a machine-readable medium including computer readable program code therein for tagging results of an XML query over a relational database as per claim 45, wherein said generated tagger node graph has a top-most tagger operator and a plurality of lower-most tagger operators, said calling and evaluating computer readable program code further comprising:

computer readable program code for performing:

a. starting with said top-most tagger operator, each tagger operator requesting results from inputs to said tagger operator, said request causing lower-level tagger operators connected to said inputs to be called;

b. starting with said lower-most tagger operators, each called tagger operator returning intermediate tagged results to a higher-level connected tagger operator upon evaluating said associated parse tree;

performing steps a and b until an end of said results of said XML query is reached, and

said top-most tagger operator producing tagged output XML of said results of said XML query.